Coalesce() Function in SQL

A brief explanation

Coalesce function

This function will search for and return the first Non-NULL value. There will be several examples to illustrate this feature. Coalesce can also be used to concatenate several rows into one single row.

Syntax ex.

SELECT COALESCE(FirstName, MiddleName, LastName) FROM Table

Quick Coalesce example

Try this quick **example**.

What do you notice?

```
SELECT COALESCE(NULL, NULL, GETDATE(), NULL) AS [Current Date]
```

Coalesce function

Difference between **Coalesce** and **ISNULL**

This distinction is important because these topics are frequently covered together and they both return the first Non-Null value but they are not the same.

Coalesce

1. Can handle two or more arguments
2. Standard SQL function (in the SQL standards. Not vendor specific.)
3. Highest data-type of input arguments is returned
4. If arguments are all Null an error will be returned *

ISNULL

1. Can handle only two inputs, maximum
2. T-SQL specific
3. Input data-type will be the type it returns
4. If arguments are all Null a Null will be returned*

*See following slide

Run these statements separately so that you see what happens when all arguments are Null.

When everything is Null

```
SELECT COALESCE(NULL, NULL)

SELECT ISNULL(NULL, NULL)
```

Another Example

Type this in exactly as it appears here.

You are declaring two separate variables and assigning them values.

The select statement will illustrate point number 3 on the preceding slide.

What do you notice?

Try this in a new query:

```
DECLARE @N varchar(3)=NULL, @Y varchar(5)='12345'

SELECT COALESCE(@N, @Y) Coal1, COALESCE(@Y,@N) Coal2,

ISNULL(@N ,@Y) n1, ISNULL(@Y, @N) n2
```

Open a new query and input the text to the right to create a quick table named Test.

Walk-thru

```
CREATE TABLE Test

(
ID int,
FirstName varchar(10),
MiddleName varchar(10),
LastName varchar(10)
);
```

Walk-thru

In the same query run the Insert Values one at a time.

Then select all to display the table to ensure it came across correctly.

```
INSERT INTO Test
VALUES (1,Null,’Sam’, Null);

INSERT INTO Test
VALUES (2,'Pablo','Neil', Null);

INSERT INTO Test
VALUES (3, Null, Null, Sara);

INSERT INTO Test
VALUES (4, James, Null, Davidson);


SELECT * FROM Test
```

Line 1: This will cycle through the rows of the columns and return the first Non-Null value.

Line 2: This will replace the value of the MiddleName columns that are Null with the value of the second argument which is ‘Unknown’ here.

Line 3: This line will return an error. Do you know why?

Walk-thru

- SELECT COALESCE(FirstName, MiddleName, LastName) FROM Test
- SELECT COALESCE(MiddleName, 'Unknown') AS [Changes] FROM Test
- SELECT ISNULL(FirstName, MiddleName,LastName) AS [Name] FROM Test

*What do you notice when you attempt to run the third query? (slide 4)

Coalesce function to concatenate

Using the School example db we will concatenate rows containing the city where the staff members are designated as faculty.

The declared variable could be anything but should be relevant to query.

```
USE SchoolSchedulingExample

DECLARE @STF VARCHAR(1000) /*This line sets the variable*/

SELECT @STF=COALESCE(@STF, ' ')+StfCity+';' FROM Staff WHERE
Position='Faculty'
/*Assigns variable to concat on the white space between the staff city names
but where the position is equal to faculty. The semicolon keeps the names from
running together.*/

SELECT @STF
/*Calls that which pertains to the variable*/
```

Create a quick table and try this example on your own!

When would I use this?

Suppose you needed to create an emergency employee contact list comprised of at least one of the following:

Work number, Cell Number, Home Number, Email Address

You know at least one is required but not all.

What are the immediate issues you may see arising? What if one more more of these are not filled out for each individual employee? Use of the COALESCE function will enable you to create a list with what information does exist.

When would I use this?

Suppose there is a sale for all items from a particular vendor but you need to set a value for any item from that vendor that has no set retail price.

What do you do?

This is where the COALESCE function comes in handy. You can simultaneously set the discount for existing retail prices and set up the procedure for those without.

```
SELECT ProductName, RetailPrice
  COALESCE(0.9*RetailPrice, 5) "Sale"
  FROM Products
  INNER JOIN Product_Vendors
  ON Products.ProductNumber=Product_Vendors.ProductNumber
  INNER JOIN Vendors
  ON Product_Vendors.VendorID=Vendors.VendorID
  WHERE VendName = 'Viscount'
```